Developer's Guide for

NetWare SFT III v3.11

Mirrored Server Link Drivers

MSL Specification Version 1.00

Disclaimer

Trademarks

Table of Contents

Chapter 1: NetWare SFT III Mirrored Servers

Chapter 2: MSL Driver Overview

Chapter 3: Data Structures, Tables, and Variables

Chapter 4: MSL Driver Procedures

Chapter 5: NetWare SFT III Support Routines

Appendix A: Building the MSL Driver

Appendix B: The NetWare Debugger

Appendix C: Getting Hardware Configuration Information

Appendix D: MSL Include File

Appendix E: MSL Driver Template

Preface

This document provides developers with the information necessary to write a Mirrored Server Link Driver for NetWare SFT III.

This document is divided into the following chapters.

Chapter 1 provides a general NetWare SFT III Mirrored Server overview.

Chapter 2 provides an introduction to the MSL driver and its required functions.

Chapter 3 describes data structures and variables that are used by the driver.

Chapter 4 contains detailed descriptions of the required MSL driver routines.

Chapter 5 contains detailed descriptions of the OS support routines available to the MSL driver.

Appendix A describes the process of assembling, linking, and loading an MSL driver.

Appendix B contains information on using the NetWare integrated debugger.

Appendix C describes methods used to obtain configuration information.

Appendix D contains a listing of the MSL.INC file.

Appendix E contains a listing for an MSL Driver template.

Manual Conventions

All numbers in this document are decimal unless otherwise specified. Hexadecimal numbers are identified by a trailing 'h' (i.e. FFh). Where bit fields within a byte are specified, bit 0 is assumed to be the low-order bit.

The following data types are defined:

byte	1 byte unsigned integer
char	1 byte ASCII character
offset	32-bit non-segmented address

Note that numeric fields composed of more than 1 byte can be in one of two formats: high-low or low-high. High-low numbers contain the most significant byte in the first byte of the field, the next most significant byte in the second byte, and so on, with the least significant byte appearing last. Low-high numbers are stored in exactly the opposite order. The Intel microprocessors store numbers in low-high order.